2. Schulaufgabe aus dem Fach Programmieren im 1. Schuljahr

Datum: 04.07.2022	Klasse: FSWI-1	Name:

PG1/2 von 50 Punkten

PG3 von 20 Punkten

Σ **von 70 Punkten**

Note:

Abschnitt PG1/2: PyQt

Während der Französischen Revolution wurde zur Abgrenzung gegenüber dem Julianischen Datums die Dezimalzeit eingeführt. Ein Tag besteht aus 10 Dezimalstunden. Jede Dezimalstunde hat 100 Dezimalminuten; jede Dezimalminute 100 Dezimalsekunden. Die Dauer eines Tages orientiert sich, wie auch beim Julianischen Datum, an der Dauer des heutigen SI-Tages:

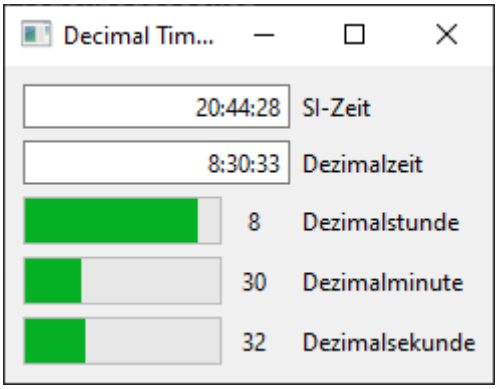
$$1\ \mathrm{d} = 24\ \mathrm{h} = 1440\ \mathrm{min} = 86\,400\ \mathrm{s}$$

Es gibt 86.400 SI-Sekunden und 100.000 Dezimalsekunden pro Tag:

- Eine Dezimalsekunde hat 86.400/100.000 = 0,864 SI-Sekunden.
- Eine Dezimalminute hat 1440/1000 = 1,44 SI-Minuten.
- Eine Dezimalstunde hat 24/10 = 2,4 SI-Stunden.

Gegeben ist der unvollständige Quelltext einer PyQt6-GUI, welche die Dezimalzeit mit verschieden Widgets darstellt. Auf dem Objekt decimalTime wird die Dezimalzeit wie folgt dargestellt:

8:00:00	für 8 Dezimalstunden, 0 Dezimalminuten, 0 Dezimalsekunden (19:12:00 Uhr SI)
8:43:75	für 8 Dezimalstunden, 43 Dezimalminuten, 75 Dezimalsekunden (20:15:00 Uhr SI)



Screenshot des QWidgets WidgetDecimalTime.

Aufgabe 1: TimeEdit

1. Erstellen Sie den Konstruktor der Klasse Time Edit, welche von QLineEdit erbt. *(3 Punkte)*
2. Setzen Sie die Eingabezeile auf schreibgeschützt und rechtsbündig über Qt.AlignmentFlag.AlignRight. *(2 Punkte)*

Aufgabe 2: DecimalTimeProgressBar

1. Geben Sie für die DecimalTimeProgressBar einen Bereich über die Variablen minimum und maximum vor. *(3 Punkte)*
2. Ändern Sie die rechte digitale Werteausgabe auf einen absoluten Wert. *(2 Punkte)*

Aufgabe 3: ThreadDecimalSecond

1. ThreadDecimalSecond erbt von QThread und enthält einen Zähler für die dezimalen Sekunden, welche seit Start des Threads vergangen sind. *(3Punkte)*
2. Über die Methode setSiSecond kann der ganzzahligen Zähler auf einen beliebigen SI-Sekundenwert gesetzt werden. Danach sendet das Signal decimalSecond mit seinem aktuellen Wert. *(5 Punkte)*
3. Die Methode run überschreibt den Prototype aus QThread mit einer Endlosschleife. *(2 Punkte)*
4. Dabei inkrementiert die Methode in passenden Abständen den Zähler und sendet das Signal decimalSecond mit dem Wert aktuellen Wert. *(5 Punkte)*
5. Beim Erreichen einer Dezimalminute wird das Signal raiseDecimalMinute ohne einen Wert gesendet. *(5 Punkte)*

Hinweis: Die Klassen DecimalMinute und DecimalHour arbeiten nach einem ähnlichen Prinzip wie ThreadDecimalSecond. Sie werden aus Zeitgründen nicht implementiert.

DecimalHour sendet nach jeder Dezimalstunde das Signal decimalHour; DecimalMinute jede Dezimalminute decimalMinute. Über den Slot steepDecimalHour wird die Dezimalstunde inkrementiert; über steepDecimalMinute die Dezimalminute.

Achtung: Die Signals und Slots der Klassen verbinden Sie erst in der letzten Aufgabe.

Aufgabe 4: TimerDecimal

1. Die Klasse TimerDecimal erbt von QTimer und ruft beim Ablauf des Timers jede Dezimalsekunde den Slot update auf. *(5 Punkte)*
2. Über einen Befehl von QDateTime erhält die Variable currentTime die aktuelle Uhrzeit als SI-Zeit. Vervollständig Sie den Quelltext um Dezimalsekunden, -minuten und -stunden aus der SI-Zeit zu errechnen. *(3 Punkte)*
3. Welche Funktion hat die vorletzte Zeile Quelltext? *(2 Punkte)*

Aufgabe 5: ThreadWatchdog
Die Methode decimalTimeToMSecond wandelt die Dezimalzeit aus Dezimalsekunden, -minuten und -stunden in SI-Millisekunden um. *(5 Punkte)*

Aufgabe 6: WidgetDecimalTime

1. Verbinden Sie nun die fehlenden Signal- and Slots. *(4 Punkte)*
2. Welche Methoden müssen Sie für ThreadDecimalSecond und ThreadWatchdog am Ende des Konstruktors aufrufen? *(1 Punkt)*

Viel Erfolg!